Enhanced Sampling Toolkit Documentation

Release 0.1a

Jeremy O. B. Tempkin

Jan 30, 2017

CONTENTS

CHAPTER

ONE

INTRODUCTION

The objective of the Enhanced Sampling Toolkit is to provide a set of tools that facilitates rapid prototyping and development of enhanced sampling algorithms in Python.

The toolkit is implemented in 100% Python. One reason for this decision is that Python is a popular language in the broader scientific community and has a wide support base for developing scientific codes. However, more importantly, a strength of the language lies in the speed at which ideas can be implemented into working code. Since rapid and flexible prototyping of new algorithms is the core priority of the toolkit, Python seems a natural choice in language. However, if the need arises in a future date, ports to other languages may be considered and integrated into the package.

In many places of the toolkit some care has been made to adhere to the powerful Python object design principles inherent in the Python data model. We find this to be a powerful feature of the Python language since it facilitates clean, "Pythonic" use of the toolkit in our applications. We are continually working to improve this aspect of the toolkit as the project develops.

In addition to rapid algorithm prototyping, we've found in it's development that the toolkit is effective in HPC environments as well. Because the underlying dynamics are executed in commonly used MD codes, the toolkit has access to the HPC features that have been optimized in these codes and can therefore leverage MPI parallelism concurrently at the algorithm level and the MD level as well as support for accelerators such as GPU's or Intel MIC cards.

1.1 Organization of the Toolkit

The toolkit consists of two fundamental layers. At the core of the toolkit is the Walker API. This API provides an standard interface between the algorithm level code and the code that executes the dynamical model. In the case of molecular dynamics, this API has been implemented around difference MD engines such as OpenMM or LAMMPS.

In the basic sense, the toolkit serves to wrap commonly used MD codes and in the process abstract the interactions between algorithm level code and the MD engine. This abstractions provides useful extensibilty in the sense that algorithm codes that are implemented in the Walker API can be reused and swapped between MD models and even entire MD codes. Furthermore, the expensive integration steps are executed in faster compiled codes and avoids some of the inefficiencies introduced in the choice of Python.

The second layer of the toolkit is the application tools. These set of tools provide a set of high-level modules that facilitate rapid programming of new Enhanced Sampling algorithms. These modules serve to act as reference implementations of core data structures and operations that are common to a set of enhanced sampling algorithms. In some cases, these tools rely on functionality provided by the Walker API.

CHAPTER

TWO

INSTALLATION

To install the Enhanced Sampling Toolkit on your local machine, download the package from the Github page.

To install the toolkit, run

```
python setup.py install
```

from the root of the toolkit package. If you plan on developing the files in the toolkit, use

```
python setup.py develop
```

which will use the state of the local files. Note that this will allow changes made locally to the package files to visible in your Python environment.